Ansible

Gérez la configuration de vos serveurs et le déploiement de vos applications (3e édition)

25 mars 2025

Table des matières

1 Introduction

Ce livre sur **Ansible** s'adresse aux **administrateurs de systèmes Unix** qui souhaitent découvrir les différentes fonctionnalités spécifiques de cet outil **DevOps** permettant la **configuration centralisée de serveurs et d'applications**. À l'aide d'exemples concrets, l'auteur apporte au lecteur les connaissances nécessaires pour bien comprendre l'intérêt de son utilisation.

Certains prérequis sur le langage **YAML** ou l'utilisation du protocole **SSH** sont un plus pour une utilisation efficace d'Ansible. Dans les premiers chapitres, l'auteur aide toutefois le lecteur à les acquérir pour qu'il puisse tirer le meilleur profit de la lecture du livre.

Les chapitres qui suivent traitent des différents mécanismes d'Ansible avec une approche de difficulté progressive. Les premiers mécanismes permettent ainsi d'administrer de façon classique les serveurs (Unix ou Windows) alors que les suivants nécessitent des notions plus avancées, notamment sur la programmation Python. Le lecteur y découvre alors comment **créer un inventaire**, comment **réinjecter des informations** provenant de sources existantes (ESX, AWS, Docker...) ou comment **créer des playbooks**. La création de **rôles Ansible** est également traitée ainsi que quelques bonnes pratiques à suivre (analyse de code et test avec Molecule à l'aide de Podman ou Docker).

À côté des notions purement orientées Ansible, certains chapitres sont consacrés au **déroulement du déploiement d'une application** MediaWiki. Le lecteur étudie ainsi les problématiques de **parallélisation des tâches**, l'introduction d'un **répartiteur de charge** Haproxy et **le lancement en séquence des opérations** permettant de réaliser les mises à jour avec un impact minimal (rolling update). L'**optimisation des tâches** sera également un point important avec la mise en place de Mitogen.

La suite du livre détaille plus particulièrement la **personnalisation d'Ansible**. La restitution d'informations (**mécanismes de callback et découverte de ARA**), l'**écriture de modules** pour la gestion d'opérations, les **filtres Jinja** ou encore la **création d'actions** sont ainsi étudiés.

Enfin, l'auteur présente dans les derniers chapitres la problématique de la **création de machines virtuelles**, classiques (via l'hyperviseur ESX/VMware/vCenter) ou dans le cloud (avec AWS), l'**utilisation de containers** Podman/Docker avec Ansible, le pilotage d'applications dans un cluster Kubernetes ainsi que la création d'un opérateur.

Auteur : Yannig PERRÉ

Administrateur système depuis de nombreuses années, **Yannig PERRÉ** est aujourd'hui spécialiste de la gestion d'applications à l'aide de conteneurs. Il associe naturellement à ce savoir-faire différents outils pour gérer les problématiques d'installation, de résilience, de scalabilité, de surveillance ainsi que de publication des applications sur Internet. Associée à sa longue expérience du monde open source, cette expertise lui perme de transmettre aux lecteurs des livres réellement efficaces sur la mise en œuvre d'Ansible, Kubernetes ou encore Prometheus et Grafana.

2 Avant-propos : D'où vient le terme DevOps ?

Au début de l'informatique, les applications étaient simples et pouvaient être gérées directement par les personnes en charge du développement.

Avec le temps, les applications se sont complexifiées et, afin d'organiser le travail et de rationaliser les coûts, des équipes ont été créées avec des spécialisations différentes. Ce découpage des tâches a débouché sur l'organisation traditionnelle que l'on retrouve actuellement en informatique : d'un côté, les développeurs (devs) et d'un autre les exploitants ou opérateurs (ops est une contraction courante pour désigner les opérateurs en anglais).

Ces équipes ont des objectifs antagonistes. Si vous prenez le cas de l'exploitation des plates-formes, pour l'exploitation, il s'agira d'un point indispensable pour assurer la pérennité de l'application. Pour un développeur, il s'agit au mieux d'un aspect sans intérêt, au pire d'une perte de temps.

Inversement, les changements seront mal perçus par les équipes d'exploitation puisque source d'instabilité, alors que pour les développeurs il s'agira de l'essence même de leur travail.

L'arrivée de l'agilité en entreprise va encore exacerber ce conflit. En effet, les équipes de développement sont déjà confrontées à l'application de ces méthodes, ce qui sera beaucoup plus rare du côté de l'exploitation.

Dans ce contexte, la mouvance DevOps a émergé afin de réconcilier ces équipes. Le terme lui-même est la contraction de Dev et Ops et a été inventé par Patrick Debois en octobre 2009.

3 Les premiers produits DevOps

Les premiers produits répondant aux impératifs de flexibilité de cette nouvelle organisation étaient déjà existants sur le marché. On peut citer notamment des produits comme Puppet, Chef, Salt ou BladeLogic. Malgré leur qualité indéniable, ces produits avaient tout de même quelques défauts :

— Ils demandaient un savoir-faire très spécifique.
— Il fallait absolument installer des agents sur les serveurs à gérer.
— Il n'est pas possible de gérer l'installation initiale de l'agent.
— Enfin, les méthodes d'administration classiques à base de SSH étaient complètement ignorées.

Pour résumer, vous deviez installer des agents supplémentaires, une infrastructure permettant de les gérer, et vous deviez reprendre totalement votre savoir-faire dans un langage très orienté développeur.

Ansible est arrivé un peu après ces pionniers pour répondre justement à ce besoin de remettre les méthodes classiques au goût du jour. En effet, avec Ansible, vous pourrez vous appuyer sur vos méthodes d'exploitation existantes (SSH, WinRM, Docker, API, etc.) :

— Pas d'agent à installer sur les machines distantes, il faut juste un interpréteur Python.
— Pas d'infrastructure à installer pour gérer les machines, vous pouvez tout faire depuis un poste quelconque avec les accès ad hoc.

4 Cibles et objectifs de l'ouvrage

Ansible est un logiciel qui s'adresse à des personnes disposant de connaissances en administration système Unix ou Windows (notamment dans tout ce qui est gestion de clé SSH) et ayant optionnellement un background de développeur Python.

Néanmoins, l'objectif du livre est de pouvoir être abordé par tout un chacun avec des premiers chapitres relativement simples à aborder pour arriver sur des concepts plus poussés.

5 Prérequis techniques et ressources documentaires

5.1 Prérequis techniques

Afin de pouvoir travailler, vous aurez besoin d'au moins une machine Linux (celle faisant tourner Ansible) ainsi que plusieurs machines virtuelles et/ou conteneurs sur lesquels vous aurez la main.

Dans le cas où la personne serait bloquée sous Windows, le mécanisme de WSL permettra à l'utilisateur de pouvoir lancer des programmes Linux depuis Windows.

5.2 Ressources documentaires

Ce livre vient avec un ensemble d'exemples permettant au lecteur de se mettre en situation (installation de serveur Apache, MySQL, création de VM, etc.). Le code source des exemples est disponible pour le téléchargement et est regroupé par chapitre de la manière suivante :

— Soit par sous-répertoires chapitre-XX dans le dépôt `https://github.com/EditionsENI/ansible`.
— Soit par archives sous la forme chapitre-XX.tar.gz sur la page du livre sur le site des Éditions ENI.

Ces fichiers sont également disponibles sous la forme d'un dépôt git disponible à l'adresse suivante : `https://github.com/EditionsENI/ansible`.

Ansible est un produit qui évolue toujours. N'hésitez donc pas à suivre les dernières évolutions en vous abonnant au groupe de discussion à l'URL ci-après : `https://groups.google.com/forum/#!forum/ansible-project`.

Vous pouvez également vous rendre sur la page GitHub du projet : `https://github.com/ansible/ansible`.

Le site `http://docs.ansible.com/` est également une source d'informations notamment sur 'utilisation des modules Ansible.

6 Présentation générale

Ce livre est une seconde édition et est composé de dix-sept chapitres. Chaque chapitre évoque une fonctionnalité spécifique d'Ansible et peut être approché de façon autonome si vous avez déjà quelques connaissances sur le produit. Vous aborderez tous les aspects du produit en passant par l'installation d'Ansible, la création de playbooks simples pour enfin déboucher sur l'écriture de plugin en Python.

Les exemples de ce livre ont été testés sur des versions récentes de CentOS 7 et Ubuntu. Ces machines virtuelles tournent sous VirtualBox. Vous pouvez bien sûr utiliser n'importe quel mécanisme de virtualisation (ESX, XenServer, OpenStack, Amazon).

À noter que dans le cas où vous auriez des difficultés à vous fournir en machines virtuelles classiques, et si vous êtes habitué à l'utilisation de Docker ou Podman, plusieurs chapitres abordent différentes techniques permettant de simuler une infrastructure classique à l'aide de conteneurs.

Au niveau des nouveautés de cette édition, l'accent a surtout été porté sur les sujets suivants :

— Un nouveau chapitre dédié à l'administration de machines Windows.
— Un chapitre dédié à l'optimisation des lancements de playbook ainsi que la découverte de Mitogen.
— Les techniques de tests des playbooks créés avec Ansible.
— Enfin le pilotage de ressources dans un cluster Kubernetes avec la création d'un opérateur.

6.1 Les prérequis

Les chapitres **Démarrer avec Ansible** et **Utilisation d'Ansible** sont là pour aider le lecteur à acquérir certains prérequis, comme par exemple le langage YAML, l'utilisation de SSH et notamment la génération et les échanges de clé. L'utilisation de Git sera également abordée afin de parler de quelques bonnes pratiques à adopter sur l'écriture des playbooks Ansible.

6.2 L'utilisation d'Ansible

Les chapitres **Découverte de l'inventaire**, **Inventaires : notions avancées**, **Fonctionnement d'un playbook**, **Introduction à la notion de rôles** et **Playbooks, rôles et notions avancées** sont là pour aborder les différents mécanismes d'Ansible sans avoir à connaître ses rouages internes. Vous y trouverez comment créer un inventaire Ansible (chapitre **Découverte de l'inventaire**), réinjecter des informations provenant de sources existantes ESX, AWS, Docker, etc. (chapitre **Inventaires : notions avancées**), créer des playbooks (chapitres **Administration Windows**, **Fonctionnement d'un playbook**, **Introduction à la notion de rôles** et **Playbooks, rôles et notions avancées**) ainsi que des rôles Ansible (chapitres **Introduction à la notion de rôles** et **Playbooks, rôles et notions avancées**) ainsi qu'à l'optimisation (chapitre **Stratégie d'exécution et optimisation**).

Les chapitres **Ansible : virtualisation et cloud** et **Tester Ansible avec Docker** font le tour des modules à disposition du lecteur pour gérer les problématiques de création de machines virtuelles ESX ou AWS (chapitre **Ansible : virtualisation et cloud**) ou de conteneurs (chapitre **Tester Ansible avec Docker**). Un dernier chapitre sera consacré à l'utilisation d'Ansible avec Kubernetes et notamment l'écriture d'un opérateur avec Ansible (**Pilotage de Kubernetes à l'aide d'Ansible**).

6.3 La personnalisation d'Ansible

Enfin, les chapitres **Sortie Ansible et centralisation**, **Écriture de modules**, **Écriture de filtres Jinja et mécanismes de lookup** et mécanisme de lookup et **Les actions Ansible** sont là pour comprendre comment fonctionne Ansible. La restitution d'informations (mécanisme de callback) dans le chapitre **Sortie Ansible et centralisation**, l'écriture de modules pour la gestion d'opérations dans le chapitre **Écriture de modules**, l'écriture de filtres Jinja dans le chapitre **Écriture de filtres Jinja et mécanismes de lookup** et enfin la création de plugins action dans le chapitre **Les actions Ansible** sont présentés.

6.4 Conventions employées

Les conventions suivantes sont employées dans le livre :

Caractères gras : désignent un fichier ou mettent en évidence certains passages.
`Caractères à chasse fixe` : utilisés pour les exemples de code dans le texte.
`$ ./commande.sh` : désignent une commande à lancer.

Le livre contient également énormément d'exemples de code source sous la forme suivante :

```
- name: "Create docker network"
  hosts: localhost
  gather_facts: no
  tasks: []
```

7 Démarrer avec Ansible : Objectifs du chapitre et prérequis

12

7.1 Contexte et prérequis

Ce chapitre aborde l'installation d'Ansible et quelques prérequis de sécurité autour de SSH.

Par la suite, vous devrez disposer de quelques prérequis notamment :

— une machine Linux avec un interpréteur Python 2.7 minimum (recommandé Python 3.5) ;

— être administrateur de cette dernière ou avoir des droits suffisants pour lancer des commandes.

L'installation d'Ansible est possible sur tous les types d'Unix du marché et notamment sous Mac OS X ou autre BSD. Attention toutefois aux effets de bord qui peuvent exister.

Il est possible de réaliser une installation d'Ansible sous Windows en faisant appel à Cygwin ou via le Linux Subsystem de Microsoft. Toutefois, les résultats peuvent varier. Si vous n'avez pas d'autre choix, il existe des solutions basées sur des machines virtuelles, ce qui aura l'avantage d'éviter les mésaventures.

Dans le cas où vous auriez des difficultés à vous fournir en VM, il est possible de passer par d'autres solutions et notamment des conteneurs Docker. Cette technique est abordée dans l'avant-dernier chapitre de ce livre.

7.2 Version de Python

Le support de Python 2 s'est arrêté depuis le 31 décembre 2019. En conséquence, il est fortement recommandé que la machine qui contrôle les déploiements utilise une version Python 3.5 ou supérieure. C'est le cas de toutes les distributions récentes Red Hat, Debian ou Ubuntu.

7.3 Environnement de travail

Par la suite, le livre fera mention d'une machine rec-apache-1. Cette dernière représente une machine qui servira pour réaliser les différents tests. Charge au lecteur de changer ce nom par celui d'une machine à sa disposition.

À noter que si vous n'avez pas de DNS, il est tout à fait possible de passer par une adresse IP.

7.4 Fichiers téléchargeables

Vous pouvez récupérer les exemples des répertoires inventaires et variables sur le repository GitHub suivant : `https://github.com/EditionsENI/ansible`

Vous pouvez également récupérer ces fichiers dans l'archive **chapitre-01.tar.gz** depuis la page Informations générales.

8 Installation d'Ansible

15

8.1 Contexte

L'installation d'Ansible peut se faire de plusieurs manières :

— Par l'intermédiaire des packages de votre système.

— À l'aide de l'outil pip de Python (éventuellement combiné avec virtualenv).

— Par l'utilisation des archives contenant le code source d'Ansible.

— Ou enfin, en interprétant directement le code source en provenance de Git.

8.2 Installation derrière un proxy

Si vous êtes derrière un proxy, il faudra exporter les variables `http_proxy`, `https_proxy` et `ftp_proxy` avec la valeur suivante : `http://mon-proxy:mon-port`. Dans le cas où il faudrait vous identifier, il faudra changer la déclaration de votre proxy pour la valeur suivante : `http://identifiant:mdp@mon-proxy:port`.

Ces variables sont valables aussi bien pour les utilitaires `apt` (Debian), `yum` (Red Hat) ou `pip` (packaging propre à Python). Ci-dessous les instructions à lancer dans votre terminal pour cette prise en compte :

```
$ export http_proxy=http://proxy:3128
$ export https_proxy=http://proxy:3128
$ export ftp_proxy=http://proxy:3128
```

8.3 Installation via les packages système

Il existe deux grandes familles de packages dans les distributions Linux : `apt` pour les dérivés de Debian/Ubuntu et `yum` (RHEL, CentOS ou Fedora).

8.3.1 Installation sous Debian/Ubuntu

Sous Debian, l'installation peut se faire à l'aide d'`apt` suivi de l'option install et du nom du package. Dans le cas d'Ansible, la commande à lancer sera la suivante :

```
$ sudo apt install ansible
```

Vérifier que la version récupérée est suffisamment récente (cf. section Vérification de la version d'Ansible).

8.3.2 Installation sur RHEL, CentOS ou Fedora

Avec `yum`, elle se fera en deux étapes :

Activation des sources EPEL (*Extra Package Enterprise Linux*)

```
$ sudo yum install epel-release
```

Installation d'Ansible

```
$ sudo yum install ansible
```

Au moment de l'écriture de ces lignes, Ansible n'est pas dans les sources officielles de Red Hat, d'où sa présence dans les sources EPEL. En revanche, vous disposerez de la dernière version stable.

8.4 Installation via pip

Dans le cas où le package que vous obtenez par votre système est trop ancien (ce qui peut arriver avec Debian par exemple) ou si vous voulez faire appel à un autre mécanisme de packaging (Gentoo, Slackware, etc.), il est également possible de passer par l'utilitaire `pip` (`pip3` pour utiliser la version de Python 3).

Sous Debian, l'installation de `pip3` se fait à l'aide de la commande `apt` suivie de l'option `install` et du nom du package python-pip3 :

```
$ sudo apt install python3-pip
```

Une fois pip installé, il est possible de lancer l'installation d'Ansible avec la commande suivante :

```
$ sudo pip install ansible
```

Dans le cas où vous auriez déjà une version installée que vous voudriez mettre à jour, il faudra ajouter l'option `-upgrade` :

```
$ sudo pip install --upgrade ansible
```

Sous Red Hat ou CentOS, il est également possible d'installer Ansible par pip. Pour cela, il faut : Installer pip avec `yum` :

```
$ yum install python3-pip
```

L'instruction fonctionne de la même façon avec `dnf` :

```
$ dnf install python3-pip
```

Lancer l'installation d'Ansible avec pip :

```
$ pip3 install ansible
```

Auparavant Ansible était également disponible sous forme de paquets. Ces derniers ne sont plus mis à jour depuis 2021 avec la version 2.9. Cette méthode d'installation est donc déconseillée.

8.5 Utilisation de virtualenv

Une dernière méthode d'installation d'Ansible (qui sera privilégiée par la suite) est de passer par l'outil `virtualenv` de Python. Ce mécanisme offre plusieurs avantages :

— Il permet de faire cohabiter très facilement plusieurs versions d'Ansible.
— Il utilise le mécanisme de packaging pip.
— Il permet de réaliser l'installation sans les droits administrateur.

En fonction du type de distribution, l'installation de virtualenv diffère légèrement. Ci-dessous la méthode pour RHEL/CentOS/Fedora :

```
$ yum install python3-virtualenv
```

Ci-dessous celle à utiliser pour l'installer sous Debian/Ubuntu :

```
$ apt install python3-virtualenv virtualenv
```

Ceci fait, il faut ensuite créer un environnement Python virtuel. Pour cela, lancez la commande `virtualenv` suivie des options suivantes :

— `-python=python3` pour spécifier la version de Python à utiliser ;
— **/tmp/ansible** pour spécifier le répertoire de travail.

Ci-dessous un exemple de lancement avec la version 3 de Python ainsi que le répertoire de travail /tmp/ansible :

```
$ virtualenv --python=python3 /tmp/ansible
```

Cet outil s'occupera alors de déposer une copie de travail indépendante dans le répertoire **/tmp/ansible**. Ci-dessous un exemple d'exécution de cette commande :

```
Running virtualenv with interpreter /usr/bin/python3
Using base prefix '/usr'
New python executable in /tmp/ansible/bin/python3
Also creating executable in /tmp/ansible/bin/python
Installing setuptools, pip, wheel...done.
.
```

Ceci fait, reste à "activer" cet environnement à l'aide du script `bin/activate` présent dans le sous-répertoire de l'environnement virtuel. Pour cela, il est nécessaire de le sourcer dans l'environnement de l'utilisateur. Ci-dessous un exemple avec l'environnement se trouvant dans **/tmp/ansible** :

```
$ . /tmp/ansible/bin/activate
```

Un bon moyen de savoir si l'on utilise un environnement virtuel est de regarder où se trouve le binaire Python avec la commande type. Ci-dessous le résultat pour l'environnement **/tmp/ansible** :

```
$ type python
python is /tmp/ansible/bin/python
```

L'interpréteur Python est bien dans le répertoire **/tmp/ansible**. L'installation d'Ansible quant à elle peut se faire avec pip comme vous l'avez vu dans la section **Installation via pip**.

FIGURE 1 – Cadenas et clé

8.6 Vérification de la version d'Ansible

Vous avez installé Ansible ? Vous pouvez maintenant vérifier la version à l'aide de la commande suivante :

```
$ ansible --version
```

Ci-dessous un exemple de sortie de cette commande (ici avec la version 2.9.2) :

```
ansible [core 2.12.10]
config file = None
configured module search path = ['/home/yannig/.ansible/plugins/modules', '...']
nsible python module location =
/home/yannig/.local/lib/python3.10/site-packages/ansible
ansible collection location =
/home/yannig/.ansible/collections:/usr/share/ansible/collections
executable location = /home/yannig/.local/bin/ansible
python version = 3.10.7 (main, Nov 24 2022, 19:45:47) [GCC 12.2.0]
jinja version = 3.1.2
libyaml = True
```

9 le protocole SSH

23

9.1 À propos de SSH

Ansible est principalement conçu pour gérer des machines à l'aide du protocole SSH ou via des commandes lancées en local. Il est également possible de gérer d'autres types de machines, comme par exemple des systèmes Windows, des conteneurs Docker ou encore via des mécanismes d'isolation (chroot ou jail).

Même s'il est possible de passer par des mots de passe pour se connecter aux machines Linux, il est fortement recommandé de passer par des clés SSH. La suite sera consacrée à la génération des clés SSH et à leur propagation sur les machines à administrer.

9.2 Clé publique et clé privée

La première chose à faire si vous n'en avez pas sera de générer une clé. Cette dernière est constituée de deux parties : la clé privée et la clé publique. Une propriété importante vient du fait qu'il est très facile d'obtenir la clé publique depuis la clé privée, mais qu'il est très difficile d'obtenir la clé privée à partir de la clé publique.

La clé privée doit rester à tout prix secrète. En effet, elle permet à celui qui la possède de se connecter à vos machines. La clé publique, quant à elle, peut être connue de tous étant donné qu'elle ne sert qu'à donner des droits de connexion à un serveur.

Ce couple de fichiers peut être comparé à un cadenas et sa clé : le cadenas ouvert représente la clé publique et la clé du cadenas représente la clé privée. Un cadenas fermé protège quelque chose et ne peut être ouvert que si on dispose de la clé du cadenas. Enfin, il est plus simple d'ouvrir un cadenas avec sa clé plutôt qu'en le cassant.

L'information est protégée par la clé publique. Le résultat ne peut être récupéré qu'avec la clé du cadenas.

Pour continuer avec cette analogie, tout le monde peut attacher un vélo en se servant d'un cadenas (même si on n'en possède pas la clé). En revanche, une fois le verrou posé, il sera beaucoup plus simple de l'ouvrir avec la clé plutôt que sans.

9.3 Génération de la clé

La génération de la clé est déclenchée par la commande `ssh-keygen`. On peut lui passer les options suivantes :

— Le type de clé à générer (rsa ou dsa) avec `-t [rsa|dsa]`.

— L'emplacement où générer la clé avec `-f <emplacement-clé>`.

— Une passphrase (phrase secrète) pour protéger la clé avec l'option `-N`.

— Éventuellement la longueur de la clé (`-b 2048` pour une clé de 2 048 bits).

Si vous ne passez pas de paramètres à la commande, `ssh-keygen` produira une clé RSA et la commande demandera les choses suivantes :

— L'emplacement de la clé (par défaut **HOME/.ssh/id**$_{r}sa$).

— **Une passphrase et la confirmation de la passphrase.**

Ci-dessous un exemple de création de clé :

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/deploy/.ssh/id_rsa):
Created directory '/home/deploy/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/deploy/.ssh/id_rsa.
Your public key has been saved in /home/deploy/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:vmP857AGZcX3DuH4U5z87ILXE6Y1bsebEnXhiSr459Y deploy@travail
The key's randomart image is:
+---[RSA 2048]----+
|           .     |
|            o o. |
|           . ++++|
|          o ..o*=|
|        So  ...=o|
|       o.. . .o=+|
|       .o.o ..*=o|
|        +o.=oEo+*|
|       ..+*+..o++|
+----[SHA256]-----+
```

9.4 Étapes de l'authentification

Il est important de comprendre comment fonctionne l'authentification par clé. Pour cela, la commande SSH procède à un certain nombre d'opérations :

— Vérification de la signature du serveur distant. Si ce dernier n'est pas connu, l'utilitaire ssh proposera de stocker la chaîne présentée par le serveur.

— Récupération des clés privées SSH présentes dans le répertoire .ssh (fichiers id$_{r}saouid_{d}sa)del'utilisateuretvrificationdesdr$

— Présentation des clés aux serveurs distants. Si une clé correspond à une entrée dans le fichier **/.ssh/authorized**$_{k}eysdistant$

— **Le client résout le challenge (c'est d'ailleurs à ce moment qu'il faut saisir la passphrase de la clé SSH) et le renvoie au serveur : l'utilisateur est authentifié.**

9.5 Parc important de machines ou hébergement dans le cloud

Dans le cas de l'administration d'un parc important de machines (ou changeant souvent de clé SSH comme dans le cas de machines hébergées dans le cloud), il n'est pas faisable de maintenir la liste des signatures de machines distantes.

La désactivation de ce mécanisme se fait à l'aide des options SSH suivantes :

— Désactivation de la vérification stricte des clés SSH des machines distantes (StrictHostKeyChecking no).

— Stockage des signatures de machines dans le fichier /dev/null (UserKnownHostsFile /dev/null).

Cette configuration se fait en alimentant le contenu du fichier **/.ssh/config**. Ci-dessous le contenu de ce fichier avec ces deux options :

```
StrictHostKeyChecking no
UserKnownHostsFile /dev/null
```

9.6 Échange de clé par mot de passe

Dans la suite de l'exercice, les connexions SSH se feront avec l'utilisateur `root`. Dans le cas où il faudrait faire appel à un autre utilisateur (`deploy`, `admin`, etc.), faites l'échange de clé avec celui-ci. Le chapitre suivant sera consacré à gérer l'escalade de droit pour passer super utilisateur avec des mécanismes tel que sudo.

Par la suite, la machine **rec-apache-1** sera administrée par clé SSH et vous disposerez du mot de passe de l'utilisateur `root`. Par conséquent, l'échange de clé se fera avec l'outil `ssh-copy-id` suivi du nom de la machine. Cet outil a pour fonction de prendre la clé publique SSH pour la déposer automatiquement sur la machine distante dans le fichier **/.ssh/authorized**$_{k}ys$.

Il est également possible de faire précéder le nom de l'utilisateur avec lequel se connecter en utilisant une arobase ('@'). Ci-dessous un exemple d'échange de clé avec l'utilisateur `root` de la machine rec-apache-1 :

```
$ ssh-copy-id root@rec-apache-1
```

Dans le cas où la communication se ferait avec l'utilisateur `deploy`, la commande serait la suivante :

```
$ ssh-copy-id deploy@rec-apache-1
```

Lors de la première connexion à une machine, `ssh-copy-id` demandera de faire confiance à la signature de cette dernière (sauf en cas de désactivation de cette vérification comme vu au cours du chapitre précédent) :

```
$ ssh-copy-id root@rec-apache-1
The authenticity of host 'rec-apache-1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:QPjmrxfQQERCRQ0SuYzBHvIdP+/1aOEQibnIkFoix2I.
Are you sure you want to continue connecting (yes/no)? yes
```

Le stockage de la signature se fait en entrant "yes". Les lignes suivantes apparaissent pour indiquer la clé trouvée et la recopie qui va être réalisée :

```
Warning: Permanently added 'rec-apache-1' (ECDSA) to the list of
known hosts.
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/deploy/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new
key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if
you are prompted now it is to install the new keys
```

Saisissez le mot de passe de l'utilisateur (ici root) :

```
root@rec-apache-1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@rec-apache-1'"
and check to make sure that only the key(s) you wanted were added.
```

La clé est maintenant en place. Comme le message vous y invite, il est maintenant possible de se connecter avec la commande `ssh <utilisateur>@rec-apache-1` pour s'assurer que l'échange de clé s'est bien passé :

```
$ ssh 'root@rec-apache-1'
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-79-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

9 paquets peuvent être mis à jour.
0 mise à jour de sécurité.

Last login: Thu Jun  8 19:43:56 2017 from 127.0.0.1
```

L'échange de clé est effectué, il est maintenant possible de réaliser les premiers tests avec Ansible.

9.7 Échange de clé sans mot de passe

L'échange de clé par mot de passe a été abordé. Le problème de cette méthode et qu'il n'est pas toujours possible d'avoir accès au sacro-saint mot de passe. Il est même probable que si vous l'obtenez, la configuration par défaut des machines empêchera la connexion directe par mot de passe (option `PermitRootLogin prohibit-password` dans le fichier /etc/ssh/sshd$_{configparexemple}$).

Dans ce cas, l'échange de clé devra être réalisé manuellement. Ci-dessous les opérations à dérouler dans pareil cas :

- **Récupérez le contenu du fichier /.ssh/id$_{rsa.pubsurlamachineAnsible}$.**
- **Connectez-vous sur la machine distante.**
- **Créez un répertoire .ssh avec des droits restreints à l'utilisateur :**

```
$ mkdir -p ~/.ssh
$ chmod 700 ~/.ssh
```

- **Créez un fichier authorized$_{keysavecgalementdesdroitsrestreints}$:**

```
$ touch ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

Déposez la clé dans le fichier authorized$_{keys}$.

9.8 Gestion d'une passphrase avec Ansible

Comme évoqué plus tôt, il est possible de protéger sa clé avec une passphrase. En effet, si ce n'est pas le cas, une personne malveillante pourrait récupérer la clé et en faire ce qu'elle veut.

D'un autre côté, il faut communiquer la clé à Ansible ce qui implique de saisir la passphrase à chaque premier lancement. Attention de bien faire le ratio entre contrainte et sécurité.

Ci-dessous un exemple de lancement d'Ansible avec une clé SSH protégée par passphrase :

```
ansible -i rec-apache-1.inv -m ping rec-apache-1
Enter passphrase for key '/home/deploy/.ssh/id_rsa':
rec-apache-1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Pour éviter d'avoir à la saisir tout le temps, il est possible de passer par l'agent SSH de la session graphique (si vous utilisez KDE ou Gnome) ou en lançant un agent SSH (dans le cas où vous seriez connecté sur un serveur distant avec une session en mode texte).

Pour ajouter la clé à l'agent, il faut passer par la commande `ssh-add` suivie du chemin vers la clé SSH. La commande demandera alors de saisir la passphrase. Ci-dessous un exemple d'ajout de clé SSH :

```
$ ssh-add .ssh/id_rsa
Enter passphrase for .ssh/id_rsa:
Identity added: .ssh/id_rsa (.ssh/id_rsa)
```

Dans le cas où vous obtenez le message *Error connecting to agent : No such file or directory*, il vous faut lancer un agent SSH avec la commande `eval $(ssh-agent)`. Ci-dessous un exemple de lancement de cet agent :

```
$ eval $(ssh-agent)
```

Cette commande doit retourner la sortie suivante :

```
Agent pid 20811
```

Dans le cas où Ansible est lancé depuis une application tierce, ces clés devront être gérées par un mécanisme externe. Dans le cas de Jenkins, le plugin SSH Agent Plugin peut être utilisé. Dans d'autres cas, il n'y aura pas forcément de solutions miracles. Il faudra alors se passer de ce mécanisme.